

# Fast and Accurate Approximation to Kriging Using Common Data Neighborhoods

Maria Vignsnes<sup>1</sup> · Odd Kolbjørnsen<sup>2</sup> · Vera Louise Hauge<sup>1</sup>  ·  
Pål Dahle<sup>1</sup> · Petter Abrahamsen<sup>1</sup>

Received: 21 December 2015 / Accepted: 15 November 2016 / Published online: 2 January 2017  
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** Unknown values of a random field can be predicted from observed data using kriging. As data sets grow in size, the computation times become large. To facilitate kriging with large data sets, an approximation where the kriging is performed in sub-segments with common data neighborhoods has been developed. It is shown how the accuracy of the approximation can be controlled by increasing the common data neighborhood. For four different variograms, it is shown how large the data neighborhoods must be to get an accuracy below a chosen threshold, and how much faster these calculations are compared to the kriging where all data are used. Provided that variogram ranges are small compared to the domain of interest, kriging with common data neighborhoods provides excellent speed-ups (2–40) while maintaining high numerical accuracy. Results are presented both for data neighborhoods where the neighborhoods are the same for all sub-segments, and data neighborhoods where the neighborhoods are adapted to fit the data densities around the sub-segments. Kriging in sub-segments with common data neighborhoods is well suited for parallelization and the speed-up is almost linear in the number of threads. A comparison is made to the widely used moving neighborhood approach. It is demonstrated that the accuracy of the moving neighborhood approach can be poor and that computational speed can be slow compared to kriging with common data neighborhoods.

**Keywords** Kriging · Fast approximation · Sub-segments · Data neighborhood · Parallelization

---

✉ Vera Louise Hauge  
vera.louise.hauge@nr.no

<sup>1</sup> Norwegian Computing Center, P.O. Box 114, Blindern, 0314 Oslo, Norway

<sup>2</sup> Lundin Norway AS, P.O. Box 247, 1326 Lysaker, Norway

## 1 Introduction

Kriging is a method of predicting values of a random field in unobserved locations based on observed data scattered in space. Kriging requires the solution of a linear equation system the size of the number of observed data that are covered by a region or domain of interest. For large data sets, the computational cost becomes large and numerical instabilities may occur. This motivates the use of data subsets, that is, local neighborhoods of data relevant for the targeted kriging locations. One local method is the so-called moving neighborhood with a geometrically defined neighborhood that moves with the target location. Several approaches investigate suitable balance between near and far observed data to be included in neighborhoods (Cressie 1993; Chilès and Delfiner 1999; Emery 2009).

This paper presents an approximation to kriging using common data neighborhoods, which is an extension of the methodology introduced in Vignes et al. (2015). In Sect. 2, kriging and its computational steps are presented and in Sect. 3 common data neighborhoods and sub-segments are introduced. Section 4 considers the accuracy of the approximation and discusses the relationship between computation time and accuracy. The methodology is furthermore extended to adaptive data neighborhoods in Sect. 5. The kriging approximation is well suited for parallelization and results demonstrating this potential are included in Sect. 6. Finally, a generalization to other forms of kriging and the prediction error is discussed in Sect. 7.

## 2 Kriging

Consider a regular grid in a hyperrectangle (orthotope)  $\mathcal{D}$  in  $\mathbb{R}^d$ . Assume that the grid covers  $\mathcal{D}$  and contains  $N$  grid nodes. The objective is to predict a random field  $z(\mathbf{x})$  at each of the  $N$  grid nodes given  $n$  observations.

By organizing the observations of  $z(\mathbf{x})$  in a  $n$ -dimensional vector

$$\mathbf{z}' = [z(\mathbf{x}_1), z(\mathbf{x}_2), \dots, z(\mathbf{x}_n)]; \quad \mathbf{x}_i \in \mathcal{D}, \quad (1)$$

the (simple) kriging equation reads

$$z^*(\mathbf{x}) = m(\mathbf{x}) + \mathbf{k}'(\mathbf{x}) \mathbf{K}^{-1} (\mathbf{z} - \mathbf{m}), \quad (2)$$

where  $z^*(\mathbf{x})$  is the predicted value at  $\mathbf{x}$ ,  $\mathbf{k}(\mathbf{x}) = \text{Cov}\{z(\mathbf{x}), \mathbf{z}\}$ ,  $\mathbf{K} = \text{Var}\{\mathbf{z}\}$  is the kriging matrix,  $m(\mathbf{x})$  is the mean value at  $\mathbf{x}$  and  $\mathbf{m}$  is a  $n$ -dimensional vector containing  $m(\mathbf{x})$  at the observation locations.

Finding  $z^*(\mathbf{x})$  in Eq. (2) is essentially done in four steps. The first step is to establish  $\mathbf{K}$ , which is an  $\mathcal{O}(n^2)$  process. Second, calculating the Cholesky factorization of  $\mathbf{K}$  is an  $\mathcal{O}(n^3)$  process. The third step is to solve the equation system using the Cholesky factorization. This is done by calculating the location independent dual kriging weights

$$\mathbf{w} = \mathbf{K}^{-1} (\mathbf{z} - \mathbf{m}). \quad (3)$$

Solving for the weights is an  $\mathcal{O}(n^2)$  process. Inserting the dual kriging weights into Eq. (2) provides the following expression

$$z^*(\mathbf{x}) = m(\mathbf{x}) + \mathbf{k}'(\mathbf{x}) \cdot \mathbf{w}. \quad (4)$$

The fourth step is to perform this computation, which is an  $\mathcal{O}(n N)$  process.

## 2.1 Computation Time

The four main steps in solving Eq. (2) are given by the following algorithm. Kriging Algorithm

1. Assemble  $\mathbf{K}$ .
2. Cholesky factorize  $\mathbf{K}$ .
3. Solve for weights:  $\mathbf{w} = \mathbf{K}^{-1}(\mathbf{z} - \mathbf{m})$ .
4. Calculate  $z^*(\mathbf{x}) = m(\mathbf{x}) + \mathbf{k}'(\mathbf{x}) \cdot \mathbf{w}$  for every grid node.

The computation time will follow

$$T_{\text{Tot}} \approx T_{\mathbf{K}} n^2 + T_{\text{Chol}} n^3 + T_{\text{Weight}} n^2 + T_{z^*} n N, \quad (5)$$

where the  $T$ 's are time constants dependent on hardware and implementation.

The two bottlenecks are steps 2 and 4. Step 2 becomes a bottleneck for large data sets and step 4 becomes a bottleneck when the number of grid nodes,  $N$ , is huge. To limit the computation time  $n$  or  $N$  must be kept small. Keeping  $N$  small is trivial but not interesting since  $N$  is usually chosen as small as possible while still remaining an acceptable spatial resolution for  $z^*(\mathbf{x})$ . Reducing  $n$  means leaving out observations from the computations. This usually gives loss of information that can affect the quality of the result.

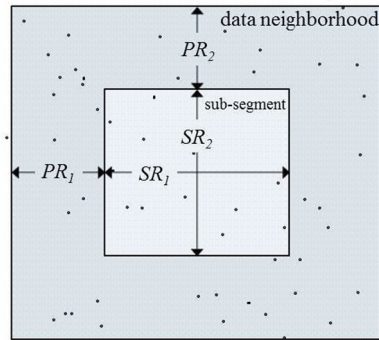
## 3 Kriging in Data Neighborhoods

A common approach to limit the number of data,  $n$ , is to use a moving neighborhood. This means that a subset of the data that is close to a grid node  $\mathbf{x}$ , is chosen when predicting  $z(\mathbf{x})$ . The number of data in the neighborhoods is usually chosen quite small ( $<100$ ) (Emery 2009). The downside of this approach is that all four steps in solving Eq. (2) must be calculated for every grid node, allowing for little reuse of computed vectors and matrices. This potentially makes the computation time very long for large grids, although the approach is highly parallelizable.

### 3.1 Common Data Neighborhoods

Instead of looking at grid node specific data neighborhoods,  $\mathcal{D}$  is divided into sub-segments where each sub-segment shares a common data neighborhood. This way the location independent weights,  $\mathbf{w}$  in Eq. (3), can be reused for all the grid nodes

**Fig. 1** Illustration of a two-dimensional sub-segment  $\mathcal{D}_i$  and its corresponding data neighborhood



inside the sub-segment. For simplicity, equally sized sub-segments  $\mathcal{D}_i$  that are hyper-rectangles in  $\mathbb{R}^d$ , are used. The sub-segments are made so that they divide  $\mathcal{D}$  into  $M$  disjoint sets.

The common data neighborhoods also take the form of hyper-rectangles in  $\mathbb{R}^d$  and contain the corresponding sub-segments. The computation time now reads

$$T_{\text{CDN}} \approx T_{\mathbf{K}} \bar{n}^2 M + T_{\text{Chol}} \bar{n}^3 M + T_{\text{Weight}} \bar{n}^2 M + T_{z^*} \bar{n} N, \quad (6)$$

where  $\bar{n}$  is the number of data in the common data neighborhoods. Comparing with Eq. (5) and noting that the Kriging Algorithm is dominated by steps 2 and 4, shows that a speed-up is obtained if

$$\bar{n}^3 M < n^3 \quad \text{and} \quad \bar{n} < n. \quad (7)$$

The challenge is, therefore, to find  $\bar{n} < n$  that gives an acceptable accuracy. To analyze this, assume that the sides of the hyperrectangle  $\mathcal{D}$  are large compared to the correlation ranges. Under this assumption, the only relevant length scales are the correlation ranges,  $\{R_1, R_2, \dots, R_d\}$ . The sub-segments and the common data neighborhoods are, therefore, chosen proportional to the correlation ranges in each direction. This is illustrated in Fig. 1 for two dimensions. The sub-segment size is  $SR_1 \times SR_2$  and the size of the data neighborhood is  $(2P + S)R_1 \times (2P + S)R_2$ , where  $S$  and  $P$  are dimensionless constants. The value of  $P$  gives the extension of the data neighborhood beyond the sub-segment, and determines the overlap between common data neighborhoods.

Let the data density and grid node density be defined as

$$n_R = n \frac{V_R}{V_{\mathcal{D}}} \quad \text{and} \quad N_R = N \frac{V_R}{V_{\mathcal{D}}}, \quad (8)$$

where  $V_{\mathcal{D}}$  is the volume of  $\mathcal{D}$  and  $V_R$  is the volume defined by the product of the correlation ranges:  $V_R = R_1 R_2 \dots R_d$ . The data density and the grid node density

are the average number of data and grid nodes in the hyper-rectangle defined by the correlation ranges of the variogram.

The number of sub-segments,  $M$ , and the average number of data in a common data neighborhood,  $\bar{n}$ , can be expressed as

$$M = \frac{V_{\mathcal{D}}}{V_R S^d} \quad \text{and} \quad \bar{n} = n_R (2P + S)^d, \quad (9)$$

so that Eq. (6) can be rewritten as

$$\begin{aligned} T_{\text{CDN}}/N \approx T_{\mathbf{K}} \frac{n_R^2 (2P + S)^{2d}}{N_R S^d} + T_{\text{Chol}} \frac{n_R^3 (2P + S)^{3d}}{N_R S^d} \\ + T_{\text{Weight}} \frac{n_R^2 (2P + S)^{2d}}{N_R S^d} + T_{z^*} n_R (2P + S)^d. \end{aligned} \quad (10)$$

This formulation allows for optimization of the computation time by selecting the value of  $S$ , for a given overlap  $P$ , independently of the number of grid cells  $N$  and number of data  $n$ . The sub-segments along the edges of  $\mathcal{D}$  will, on average, have less than  $n_R$  data points in the data neighbourhood, so Eq. (6) over-estimates the true computation time. This over-estimation becomes more pronounced as variogram ranges increase.

## 4 Finding Optimal Sub-segment Size and Common Data Neighborhoods

### 4.1 Controlling the Accuracy

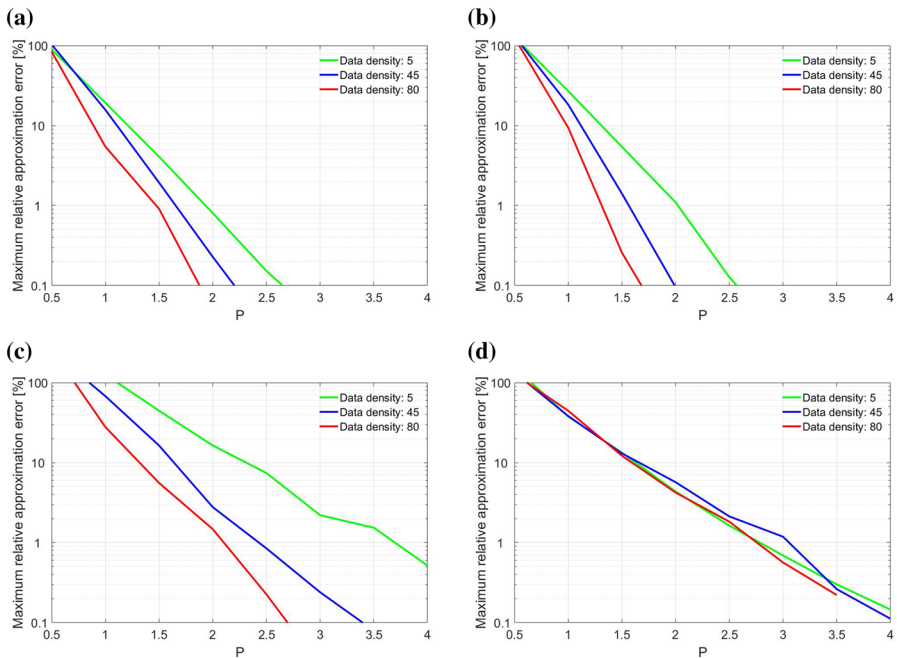
The choice of the overlap  $P$  is a compromise between acceptable accuracy and acceptable computation time. This choice is independent of, and valid for, any sub-segment size  $S$ . For a given overlap  $P$ , the computation time is optimized by selecting the optimal sub-segment size  $S$ .

In the following, accuracy is determined by the maximum relative approximation error

$$\text{MRAE} = \frac{\max_{\mathbf{x} \in \mathcal{D}} |z^*(\mathbf{x}) - z_{\text{CDN}}^*(\mathbf{x})|}{\sigma}, \quad (11)$$

where  $z^*(\mathbf{x})$  is the kriging predictor obtained using all data,  $z_{\text{CDN}}^*(\mathbf{x})$  is the kriging predictor obtained using common data neighborhoods, and where  $\sigma^2 = \text{Var}\{z(\mathbf{x})\}$  is the sill.

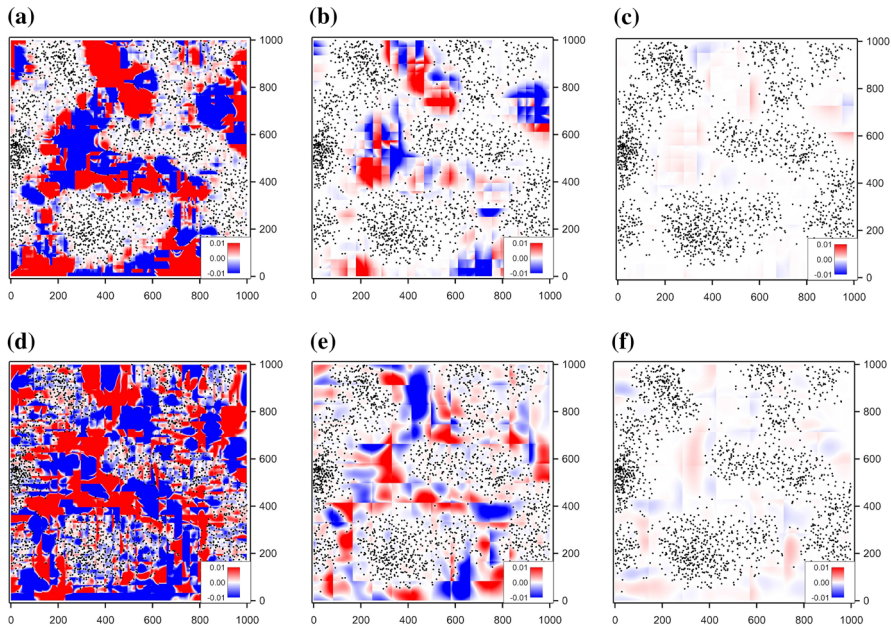
For sufficiently large overlap  $P$ , all data belong to all common data neighborhoods and the maximum relative approximation error is zero. For smaller overlaps, the maximum relative approximation error will depend on variogram type and the data density,  $n_R$ . The empirical maximum relative approximation error for a given overlap  $P$  is found by comparing kriging using common data neighborhoods with kriging using all data. Kriging is repeated 100 times using simulated sets of observation values. An estimate of the maximum relative approximation error is obtained by choosing the largest error from the 100 samples. Details are given in Appendix A.



**Fig. 2** Maximum relative approximation error versus data overlap  $P$  for different data densities and different variograms: **a** general exponential with power 1.0, **b** general exponential with power 1.5, **c** general exponential with power 1.99, and **d** spherical

Figure 2 shows the empirical relationship between the overlap  $P$  and the maximum relative approximation error for four variograms and three data densities of 5, 45 and 80. The three data densities correspond to correlation ranges 50, 150 and 200, respectively. For all variograms, an approximate log-linear relationship between the overlap  $P$  and the maximum relative approximation error can be observed. It is also observed that the maximum relative approximation error seems independent of the data density  $n_R$  for the spherical variogram but for the exponential variograms the error increase with smaller data densities. Figure 2 shows that the common data neighborhoods must extend significantly beyond one range to get a maximum relative approximation error as low as 1%. For the spherical variogram, for instance, the neighborhood must extend more than 3 ranges, which may seem counter-intuitive for a variogram that has a finite range. The reason is the relay effect (Chilès and Delfiner 1999), which is particularly strong for the spherical variogram.

Figure 3 shows examples of error maps  $(z^*(\mathbf{x}) - z_{\text{CDN}}^*(\mathbf{x}))/\sigma$  for two different variograms and three different values of overlap  $P$ . As the overlap increases, the error in regions with sparse data successively decreases. When increasing the overlap by 0.5 for the general exponential variogram, the maximum relative approximation error decreases by approximately one order of magnitude. To obtain the same decrease in maximum relative approximation error for the spherical variogram, the overlap has to be increased by 1. This is consistent with the steeper slope observed for the exponential variogram in Fig. 2.



**Fig. 3** Error maps for the exponential variogram with power 1.5 (*top row*) and for the spherical variogram (*bottom row*). Data density is  $n_R = 45$ . Overlap  $P$  and maximum relative approximation errors are **a**  $P = 0.5$ ,  $\text{MRAE} = 66\%$ , **b**  $P = 1.0$ ,  $\text{MRAE} = 5.1\%$ , **c**  $P = 1.5$ ,  $\text{MRAE} = 0.46\%$ , **d**  $P = 1$ ,  $\text{MRAE} = 21\%$ , **e**  $P = 2$ ,  $\text{MRAE} = 3.1\%$ , and **f**  $P = 3$ ,  $\text{MRAE} = 0.25\%$

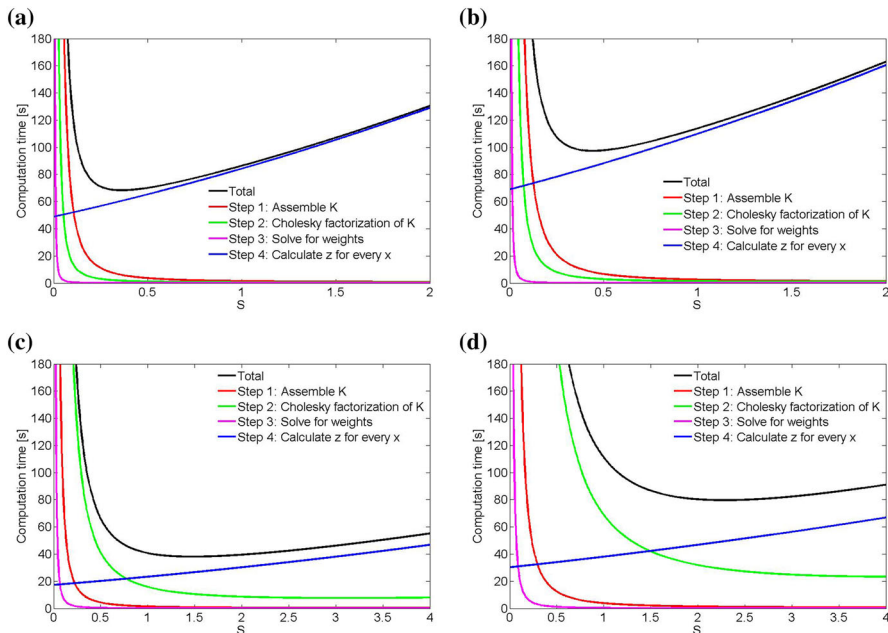
## 4.2 Minimizing the Computation Time

For a given overlap  $P$ , the sub-segment size  $S$  that minimizes the computation time, can be found by doing a one-dimensional minimization of Eq. (10). A simple half-interval search (binary search) has proved very effective.

Figure 4 shows the computation time versus sub-segment size  $S$  for two different variograms and two values of overlap  $P$  corresponding to maximum relative approximation error of 1 and 0.1% for data density  $n_R = 45$  and  $N = 10^6$  grid cells. The terms in Eq. (10) and the time constants from Table 1 are used to calculate the times in Fig. 4. The time constants have been found by direct time measurements of the separate steps in the Kriging Algorithm, and depend on implementation and hardware. This implementation is run on an Intel Xeon X5690 3.47 GHz, using the Intel Math Kernel Library for linear algebra.

The optimal sub-segment size  $S$  is found at the minimum of the total computation time (Fig. 4). When reducing the maximum relative approximation error from 1 to 0.1%, the optimal sub-segment size  $S$  increases from 0.35 to 0.45 for the general exponential variogram and from 1.5 to 2.3 for the spherical variogram. For the spherical variogram the total computation time is close to constant for a wide interval of sub-segment sizes, thus any choice of  $S$  within this interval is efficient. If, on the other hand, the sub-segment is chosen too small, the computation time will escalate. The optimal sub-segment size is higher for the spherical variogram than for the general





**Fig. 4** Computation time versus sub-segment size  $S$  for the terms included in Eq. (10), for the exponential variogram with power 1.5 (*top row*) and the spherical variogram (*bottom row*). Data density is  $n_R = 45$  and the number of grid nodes is  $N = 10^6$ . The maximum relative approximation error and overlap  $P$  are **a** MRAE = 1%,  $P = 1.6$ , **b** MRAE = 0.1%,  $P = 1.9$ , **c** MRAE = 1%,  $P = 3.1$ , and **d** MRAE = 0.1%,  $P = 4.1$

exponential. A default value for the sub-segment size  $S$ , valid for all variograms and all overlaps, is, therefore, not favorable. However, values around 1 seem acceptable in most situations.

Computing the optimal overlap  $P$  and sub-segment size  $S$  is potentially a time-consuming task. For a given variogram, however, the values of  $P$  and the corresponding accuracy can be pre-tabulated as in Fig. 2. With a value of  $P$  selected according to a required accuracy, the optimal overlap  $S$  can efficiently be found by minimizing Eq. (10). Accurate time constants in Eq. (10) are not crucial for obtaining a value of  $S$  that gives acceptable computation time, hence constants from Table 1 can be applied.

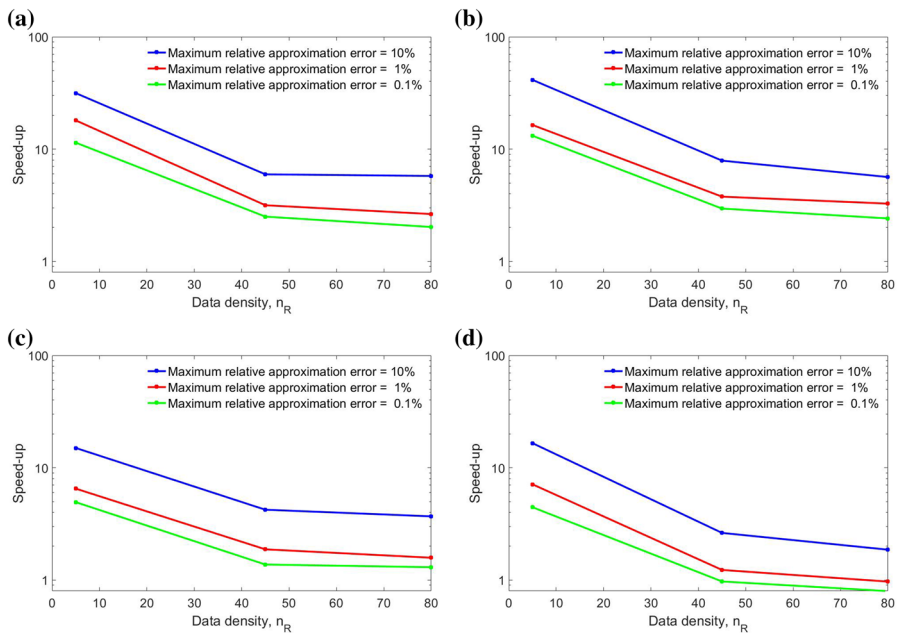
Figure 5 shows the speed-up for kriging using common data neighborhoods compared with kriging using all data. The gain is high for the smallest data densities, where computations are up to 40 times faster than kriging using all data. The speed-up is more moderate for higher data densities, in the range of 1.3–8 for the general exponential variograms. A speed-up is obtained for all cases, except when high accuracy is required, and when a spherical variogram is used and the data density is high. Anisotropic correlation functions where the principal directions of the anisotropy do not follow the grid axes, will cause the data neighborhoods to include data that are unnecessary. The best speed-ups are, therefore, obtained by rotating the grid so that it follows the anisotropy directions.



**Table 1** The time constants in Eq. (10) for different variograms

Variogram type	$T_K$	$T_{Chol}$	$T_{Weight}$	$T_{z^*}$
Spherical	6	0.028	0.56	10
General exponential, power 1.0	24	0.030	0.60	48
General exponential, power 1.5	54	0.028	0.56	106
General exponential, power 1.99	57	0.083	0.56	113

Values are given in nanoseconds ( $10^{-9}$  s)



**Fig. 5** Speed-up for various values of data density. The speed-up is relative to kriging using all data. The variograms are **a** general exponential with power 1.0, **b** general exponential with power 1.5, **c** general exponential with power 1.99, and **d** spherical

### 4.3 Comparison with Moving Neighborhood

A moving neighborhood is equivalent to using a sub-segment size of one single cell, that is, by setting  $S = N_R^{-1/d}$ . This choice of  $S$  is very inefficient, as can be illustrated by the two-dimensional case in Fig. 4a. With the optimal  $S$  of 0.35, the computation time is approximately 1 min, while the moving neighborhood approach ( $S = 0.0067$ ) takes 2.7 h.

In practice, most moving neighborhood algorithms must sacrifice accuracy by using few data in the neighborhoods, to get acceptable computation times. By using the 100 closest data points only, the computation time reduces to 14.1 min, and by reducing the number of data points to 20, the computation time drops to 2.5 min. In these

estimates the times spent on the search for the closest neighbors to all cells are 2.3 and 1.8 min, respectively. The maximum relative approximation errors for these calculations are 11 and 49%, respectively, as can be compared to the error of 1% for the 1 min common data neighborhoods approach. To illustrate this consider the prediction of depth to a geological surface where the standard error (sill) of the variogram is 20 m. A maximum relative approximation error of 1% corresponds to a maximum numerical error of 0.2 m that is acceptable in most situations. The moving neighborhood approaches would give maximum numerical depth errors of 2.2 and 9.8 m, respectively. These errors are probably too large to be acceptable in many situations.

For a comparison, the KB2D executable in GSLIB (Deutsch and Journel 1998) has been run on the same example using the 100 and 20 closest data and using the exponential variogram. This resulted in computation times of 26.8 and 1.1 min. Our implementation yielded correspondingly 8.1 and 2.0 min, of which 2.2 and 1.8 min were used for the search for closest neighbors. The KB2D algorithm is twice as fast for the example with the 20 closest data. This is probably because it uses a more efficient approach to finding neighboring data. However, for the 100 closest data the computation time is more than 3 times longer using GSLIB. This is probably due to different efficiencies in the linear algebra libraries.

## 5 Adaptive Data Neighborhoods

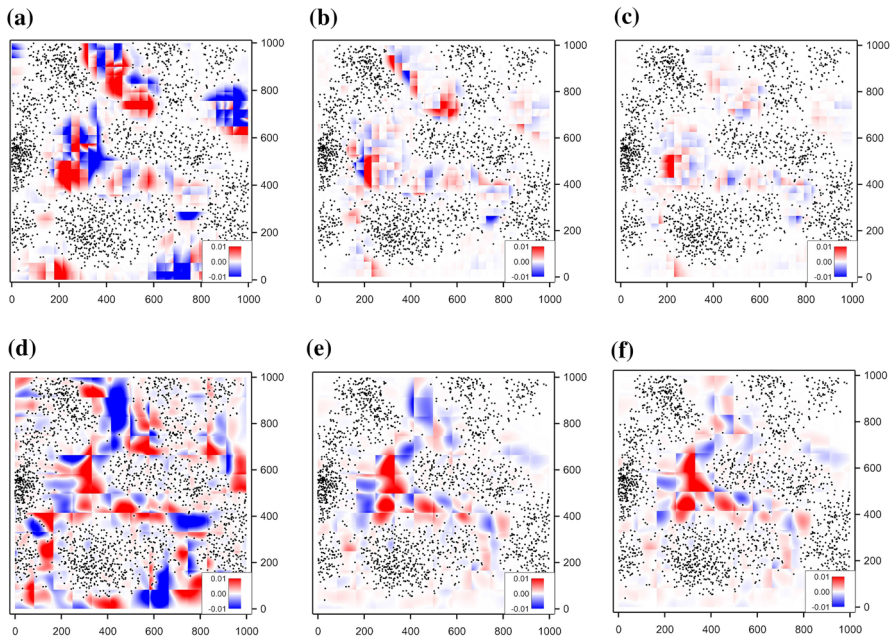
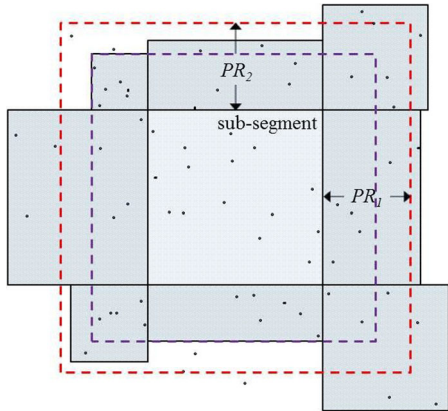
Minimizing the computation time according to Eq. (10) is performed under the assumption of uniformly distributed data, which is an unrealistic assumption for real cases. Sub-segments with number of data much larger than the average,  $\bar{n}$ , may increase the computation time significantly since the Cholesky factorization (step 2 in the Kriging Algorithm) is proportional to  $\bar{n}^3$ . On the other hand, sub-segments with the number of data much smaller than  $\bar{n}$  may give lower accuracy.

To handle local variations in data densities, adaptive data neighborhoods can be used. The size of a specific data neighborhood can be chosen such that the number of data in the neighborhood is close to  $\bar{n}$ . This is best done by choosing an overlap  $P$  for each sub-segment using a half-interval binary search. It is reasonable to limit the binary search, for instance within  $[P/2, 2P]$ , to avoid too small and too large neighborhoods.

The data may be unfavorably distributed, for instance when all data points are located on one side of the sub-segment. A further extension to compensate for this is to perform a binary search in each direction. In two dimensions this means aiming for a number of data with a minimum of  $\bar{n}/8$  in each of the eight directions. This means the data neighborhoods are only extended in the directions where there is too little data. Figure 6 illustrates the directional adaptive data neighborhoods in two dimensions. The adaptive neighborhood is the purple dashed rectangle and the common data neighborhood the red dashed rectangle.

Figure 7 shows examples of error maps using a common data neighborhood with constant size, using adaptive data neighborhood and using directional adaptive data neighborhood, for a general exponential variogram with power 1.5 and the spherical variogram. Table 2 summarizes the ranges of the overlap  $P$  and number of data in the

**Fig. 6** Example of a two-dimensional directional adaptive neighborhood where a binary search has been performed in each direction. The data neighborhood is within the gray boxes. The red dashed line outlines the common data neighborhood and the purple dashed line outlines the adaptive data neighborhood



**Fig. 7** Error maps for (a, d) common data neighborhood with constant overlap, a  $P = 1$  and d  $P = 2$ , (b, e) adaptive data neighborhoods, b  $\bar{P} = 1.14$  and e  $\bar{P} = 2.78$ , and (c, f) directional adaptive data neighborhoods, c  $\bar{P} = 1.57$  and f  $\bar{P} = 3.38$ , for the general exponential variogram with power 1.5 (top row), and the spherical variogram (bottom row). Data density is  $n_R = 45$

neighborhoods, in addition to error measures and computation times. Note that average number of data, average  $n$ , is the actual average, while the targeted values are  $\bar{n} = 216$  and  $\bar{n} = 904$  for the general exponential and spherical variogram, respectively. For the data used in this paper, there is a small decrease in maximum relative approximation error from 5.1% using common data neighborhood to 3.4% using adaptive neighborhood for the general exponential variogram with power 1.5 and data density  $n_R = 45$ . The same level of maximum relative approximation error is achieved using

**Table 2** Specifications of values of overlap  $P$ , number of data, error measures and computation times for the different data neighborhoods

	Common	Adaptive	Directional adaptive
Gen exp 1.5			
$\bar{P}$	1	1.14	1.57
$[P_{\min}, P_{\max}]$	[1, 1]	[0.54, 2.00]	[0.64, 2.00]
Average $n$	185	221	250
$[n_{\min}, n_{\max}]$	[17, 392]	[175, 230]	[175, 310]
MRAE	5.1%	3.4%	3.0%
Mean error	0.18%	0.054%	0.029%
Comp. time (s)	23	28	32
Spherical			
$\bar{P}$	2	2.78	3.38
$[P_{\min}, P_{\max}]$	[2, 2]	[1.80, 4.00]	[2.44, 4.00]
Average $n$	625	918	972
$[n_{\min}, n_{\max}]$	[226, 1019]	[710, 929]	[710, 1099]
MRAE	3.1%	3.4%	2.6%
Mean error	0.24%	0.080%	0.068%
Comp. time (s)	11	16	18

Mean error is the mean of the absolute error map

the directional adaptive data neighborhood. No decrease in maximum relative approximation error is observed for the spherical variogram. However, for both variograms, the error in regions with sparse data is smaller using an adaptive neighborhood than using common data neighborhood. This is also reflected by the decrease in mean of the error maps, by a factor of 3.3 and 6.2 for the general exponential variogram, and a factor of 3 and 3.5 for the spherical variogram. Hence, a general improvement in accuracy is achieved with the adaptive data neighborhoods, although maximum relative approximation error is hardly decreased. The computation time increases mainly due to the increased average number of data in the adaptive neighborhoods.

## 6 Parallelization

Kriging is well suited for parallelization, as the predicted values are calculated independent of each other. This also holds for sub-segments as the prediction in one sub-segment is independent of the other sub-segments. There are typically more than 100 sub-segments, and the amount of calculation for each sub-segment is large while the amount of overhead is small. The granularity is, therefore, good, and an efficient parallelization can be obtained. [Hu and Shu \(2015\)](#) present an MPI-based kriging algorithm where the grid cells to be estimated are split into a few blocks and assigned to each processor at once. This parallelization demonstrates significant speed-ups, but assumes a small number of observations. Here the OpenMP API ([OpenMP 2008](#)) is used for the parallelization. Using dynamic scheduling, a new sub-segment is assigned

**Table 3** Speed-up factors relative to using 1 thread, when predicting  $10^6$  cells conditioned to 2000 data using general exponential variogram with power 1.5 and the spherical variogram, and data density of 45

# threads	Gen exp 1.5				Spherical			
	<i>P</i>				<i>P</i>			
	0.5	1.0	1.5	2.0	1.0	2.0	3.0	4.0
2	2.0	2.0	2.0	2.0	1.9	1.9	2.0	1.9
4	3.8	4.0	4.0	3.9	3.9	3.8	3.8	3.7
8	7.2	7.6	7.7	7.6	7.1	7.1	6.9	6.2
12	10.1	11.0	11.1	11.2	10.1	9.7	9.2	8.3
24	13.4	15.3	15.4	15.0	11.9	10.5	9.6	8.5
Comp. time (s)	6.6	23.1	46.1	72.8	4.0	10.7	17.8	23.7

The computation time specifies the wall time using 1 thread

to a thread when the thread has finished predicting the previous sub-segment. Table 3 summarizes speed-up factors for different numbers of threads and different values of overlap  $P$ , predicting  $N = 10^6$  grid cells conditioned to  $n = 2000$  data. The calculations have been performed on an Intel Xeon X5690 3.47 GHz based system with 76 Gb memory and 2 CPUs, each with 6 physical cores with hyper-threading enabled, resulting in 24 threads. The table shows results for the general exponential variogram with power 1.5 and the spherical variogram. The speed-up factors are relative to using 1 thread and can be seen to scale nearly linearly with the number of threads, up to the number of physical cores. A small additional gain is obtained with hyper-threading.

In real applications, the number of data in the data neighborhoods can vary a lot. The time spent predicting each sub-segment can, therefore, differ by orders of magnitude. To avoid one thread to get stuck with a huge task at the end of the calculation, the sub-segments should be sorted with a decreasing number of data in their common data neighborhoods. This ensures an efficient parallelization where the threads finish roughly at the same time. It has been observed that this can have a huge impact in some cases.

## 7 Extension to Universal Kriging, Bayesian Kriging, Prediction Error and Conditional Simulations

Extension from simple to universal or Bayesian kriging (Omre and Halvorsen 1989) is possible by recognizing that the known trend,  $m(\mathbf{x})$ , in Eq. (2) can be replaced by a linear combination of known explanatory variables

$$m(\mathbf{x}) = \sum_{l=1}^L \beta_l f_l(\mathbf{x}). \quad (12)$$

For universal kriging the coefficients are estimated from data using generalized least squares. For Bayesian kriging the posterior distribution for the coefficients are obtained using a prior normal distribution that is updated to the posterior normal distribution by

the data. Both approaches require solving a linear equation system involving the full kriging matrix. This is only done once, however, and there is still a significant benefit in splitting the calculations into sub-segments with common data neighborhoods, as step 4 of the Kriging Algorithm typically dominates the computation time (see Fig. 4).

Simple kriging provides a prediction error often called the kriging error

$$\sigma^{*2}(\mathbf{x}) = \sigma^2 - \mathbf{k}'(\mathbf{x}) \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}), \quad (13)$$

where  $\sigma^2 = \text{Var}\{z(\mathbf{x})\}$  is the sill. Using common data neighborhoods for sub-segments is efficient when solving these equations. However, it is not possible to use the dual kriging approach since the linear equation system now depends on  $\mathbf{x}$ . This means that the Cholesky decomposition can be reused for sub-segment  $i$ , but solving the dual kriging weights in step 3 in the Kriging Algorithm must be replaced by solving the kriging weights

$$\mathbf{w}(\mathbf{x}) = \mathbf{k}'(\mathbf{x}) \mathbf{K}^{-1}. \quad (14)$$

The computation time for calculating the prediction error is typically 2–4 times longer than calculating the predictor using the dual kriging weights. Similar expressions for the kriging error exist for universal and Bayesian kriging and the similar increase in computation time applies for these methods.

The classic approach to conditional simulations (Journal and Huijbregts 1978) is to generate an unconditional Gaussian random field and condition it to data using (simple) kriging. If an efficient approach to the unconditional simulations is used, like the Fast Fourier Transform based algorithms (Dietrich and Newsam 1993; Wood and Chan 1994), the time consuming part is still the conditioning, and the suggested approach will give significant speed-ups.

## 8 Conclusions

An approximation to kriging has been proposed and tested. The idea is to divide the region of interest into rectangular sub-sets with overlapping data neighborhoods. The approximation gives significant speed-ups ( $\sim 2$ –40) depending on data density and the chosen acceptable accuracy.

The accuracy is controlled by selecting the overlap  $P$  of the common data neighborhoods. It requires time consuming simulation experiments to find a relationship between variogram type, overlap  $P$  and accuracy. Figure 2 summarizes results from such simulation experiments for some widely used variograms. These results can be used to select a reasonable value for the overlap  $P$  for many situations. For instance, accepting maximum relative approximation error of 5% can be obtained using  $P = 1.5$  for the exponential variogram and  $P = 2$  for the spherical variogram. Smoother variograms will require larger overlap  $P$  to obtain the same accuracy, especially when there are little data. Choosing the overlap  $P$  is a trade-off between accuracy and speed. For a given overlap  $P$  the sub-segment size  $S$  that minimize the computation time can be found by minimizing Eq. (10). This is a simple one-dimensional search that takes hardly any time. Figure 4 shows that a default value of  $S = 1$  gives good efficiency in most cases.

The approximation is further refined by introducing adaptive data neighborhoods, where the data neighborhoods are allowed to shrink and expand depending on the data density near the sub-segments. Adaptive neighborhoods improve the accuracy, in particular for areas of sparse data. The improved accuracy is moderate and must be weighted against the added algorithm complexity. The approximation is well suited for parallelization since the granularity of the tasks is so that the overhead is small. This is demonstrated by the almost linear scaling shown in Table 3. The speed-up from parallelization comes on top of the speed-up obtained from using sub-segments with overlapping data neighborhoods. Obtaining combined speed-ups of a factor 400 is therefore possible in situations when 5% maximum relative approximation error is acceptable.

A comparison to the widespread moving neighborhood algorithm has been made. It is shown that using sub-segments with overlapping data neighborhoods are superior, both on computation time and accuracy, when there are many data and the domain of interest is significantly larger than the volume defined by the variogram ranges. The analysis shows that moving neighborhood algorithms are poor approximations to kriging using all available data since common practice is to limit the number of data in the neighborhood to a small number.

**Acknowledgements** The paper was funded by the Research Council of Norway.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

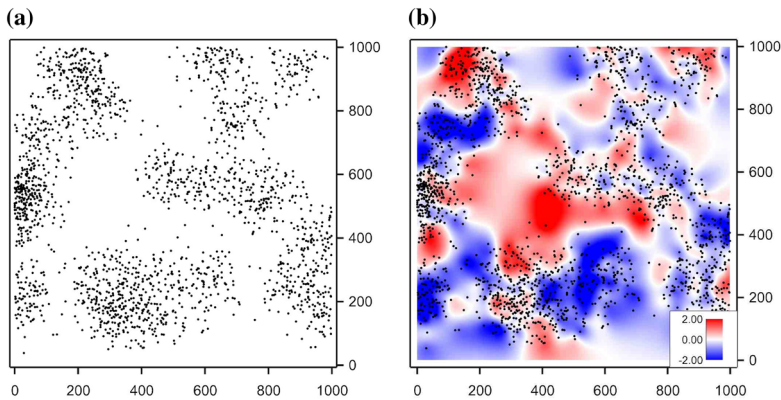
## Appendix A: Details of the Kriging Setup

Data sets of  $n = 2000$  observations are generated in the domain  $\mathcal{D}$ . The data are drawn from a zero mean Gaussian distribution with variance one and correlation determined by the chosen variogram. Figure 8a shows the fixed locations of the data observations inside the  $1000 \times 1000$  square domain  $\mathcal{D}$ . Note that some areas have clustered data, while other areas have no observations.

Four variogram types are used for the Gaussian random field. These are the general exponential variogram with power of 1.99, 1.5 and 1.0, and the spherical variogram. Data densities of 5, 45 and 80 are obtained using correlation ranges 50, 150 and 200, respectively. The side lengths of the domain  $\mathcal{D}$  are in this case 1000. Grid node densities are correspondingly 2500, 22,500 and 40,000 for the given ranges.

A total of 100 data sets from different Gaussian random fields are drawn for each combination of variogram type and range. For each of the simulated data sets, kriging is performed in a grid of  $N = 1000 \times 1000 = 10^6$  cells. Kriging using common data neighborhoods is performed using various values of overlap  $P$  and compared with kriging using all data. Based on all simulations, the relation between the overlap  $P$  and the maximum relative approximation error is established and reported in Fig. 2. The maximum relative approximation error in Eq. (11) is the maximum over all 100 simulation runs with the same variogram and same overlap. The computation times for





**Fig. 8** *Left plot* shows the locations of the 2000 data points and the *right plot* shows a kriged field, using the general exponential variogram with power 1.5 and data density of 45

the approximation using the different combinations of overlap and data densities, are compared to the computation time of kriging using all data. The achieved speed-up is reported in Fig. 5. The time constants in Table 1 are found by direct time measurements of the separate steps in the Kriging Algorithm. They depend on implementation and hardware, but our experience is that obtaining accurate constants is not critical for obtaining significant speed-up of the approximation.

One of the simulated data sets is used for the error maps in Figs. 3 and 7. Figure 8b shows the kriged field of this data set, using the general exponential variogram with power 1.5 and data density of 45.

## References

- Chilès JP, Delfiner P (1999) Geostatistics: modeling spatial uncertainty. Wiley, New York
- Cressie N (1993) Statistics for spatial data. Wiley, New York
- Deutsch CV, Journel AG (1998) GSLIB: geostatistical software library and user's guide. Oxford University Press, New York
- Dietrich CR, Newsam GN (1993) A fast and exact method for multidimensional gaussian stochastic simulations. *Water Resour Res* 29(8):2861–2869. doi:[10.1029/93WR01070](https://doi.org/10.1029/93WR01070)
- Emery X (2009) The kriging update equations and their application to the selection of neighboring data. *Comput Geosci* 13(3):269–280. doi:[10.1007/s10596-008-9116-8](https://doi.org/10.1007/s10596-008-9116-8)
- Hu H, Shu H (2015) An improved coarse-grained parallel algorithm for computational acceleration of ordinary Kriging interpolation. *Comput Geosci* 78:44–52. doi:[10.1016/j.cageo.2015.02.011](https://doi.org/10.1016/j.cageo.2015.02.011)
- Journel AG, Huijbregts CJ (1978) Mining geostatistics. Academic Press Inc., London
- Omre H, Halvorsen K (1989) The Bayesian bridge between simple and universal kriging. *Math Geol* 21(7):767–786. doi:[10.1007/BF00893321](https://doi.org/10.1007/BF00893321)
- OpenMP (2008) OpenMP application program interface version 3.0. <http://www.openmp.org>
- Vigsnes M, Abrahamsen P, Hauge VL, Kolbjørnsen O (2015) Efficient neighborhoods for kriging with numerous data. In: Conference proceedings. Third conference on petroleum geostatistics, EAGE. doi:[10.3997/2214-4609.201413660](https://doi.org/10.3997/2214-4609.201413660)
- Wood ATA, Chan G (1994) Simulation of stationary Gaussian processes in  $[0,1]^d$ . *J Comput Graph Stat* 3(4):409–432. doi:[10.1080/10618600.1994.10474655](https://doi.org/10.1080/10618600.1994.10474655)